WHAT IS CLAIMED IS:

1. A method for an application program to access an object-oriented database (OODB), the application program being one of a portfolio management application and a trade order management system (TOMS) application, the application program being associated with an operating system, the method comprising:

(a) connecting to a data repository of the database, the data repository including a plurality of memory-mapped file segments stored on at least one nonvolatile memory medium, the file segments including objects directly interconnected by memory pointers, wherein each object has an associated stored time stamp, the time stamp indicating at least one of a time when the object first appeared in the data repository and a time when the object became invalid, wherein the data in the data repository is not copied into main memory from the data repository when needed by the application program, and wherein the database is one of a portfolio management database and a TOMS database;

(b) registering a fault handler with the operating system, the fault handler associated with the data repository;

(c) catching, by the fault handler, a segmentation fault issued for an object referenced by the application program and resident in the data repository, the segmentation fault issued at an interrupt location in the application program;

(d) finding a file segment of the data repository that corresponds to the referenced object;

(e) mapping the found file segment into main memory; and

(f) restarting the application program at the interrupt location.

2. The method of claim 1, wherein the file segments include at least one portfolio

segment and price segment.

3. A method for an application program to manage memory associated with an object-oriented database (OODB) accessed by the application program, the application program associated with an operating system, the method comprising:

(a) connecting to a data repository of the database, the data repository including a plurality of memory-mapped file segments stored on at least one nonvolatile memory medium;

(b) registering a fault handler with the operating system, the fault handler associated with the data repository;

(c) catching, by the fault handler, a segmentation fault issued for an object referenced by the application program and resident in the data repository, the segmentation fault issued at an interrupt location in the application program;

(d) finding a file segment of the data repository that corresponds to the referenced object;

(e) mapping the found file segment into a main memory; and

(f) restarting the application program at the interrupt location.

4. The method of claim 3, further comprising unmapping a second file segment from the main memory before mapping the found file segment.

5. The method of claim 4, wherein file segments are unmapped according to a least-recently-used (LRU) criterion.

6. The method of claim 3, wherein the data in the data repository is not copied into the main memory from the data repository when needed by the application program and is

instead directly accessed by the application program.

7. The method of claim 3, wherein the nonvolatile memory medium is a disk.

8. The method of claim 3, wherein objects in the data repository are directly interconnected by memory pointers.

9. The method of claim 8, wherein the objects comprise C++ objects.

10. The method of claim 8, wherein a linkage in the interconnected objects includes at least one of an X node, a Y node, and a Z node.

11. The method of claim 3, wherein each object includes a virtual function pointer, the pointer pointing to a shared memory area holding virtual function tables associated with object types.

12. The method of claim 3, wherein every object in a class of the data repository points to a same predetermined shared memory address when stored.

13. The method of claim 3, wherein each object includes a pointer to itself.

14. The method of claim 3, wherein each object in the data repository has an associated stored time stamp, the time stamp indicating at least one of a time when the object first appeared in the data repository and a time when the object became invalid.

15. The method of claim 14, wherein the time stamp is stored in a header of the object.

16. The method of claim 3, wherein the application program is a portfolio management application, and the database is a portfolio management database.

17. The method of claim 16, wherein the file segments include at least one portfolio segment and price segment.

18. The method of claim 3, further comprising:

(a) upon a request by the application program to store a new object in the database, creating a segment object in the data repository;

(b) associating a segment identifier with the new object, the segment identifier being one of a default segment identifier, a segment identifier specified by the new object, and a segment identifier specified by another object that owns the new object;

(c) if a current segment file has sufficient memory for the new object, allocating memory to the new object from the current segment file;

(d) if the current segment file has insufficient space for the new object, allocating memory to the new object by extending the current segment file or creating a new segment file; and

(e) storing the new object in the allocated memory.

19. The method of claim 18, wherein memory is allocated in 16 megabyte segments.

20. The method of claim 18, wherein segment files are extended or created in 1

megabyte segments.

21. The method of claim 3, wherein the data repository is connected to a NFS (Network File System) network.

22. The method of claim 21, wherein multiple computers access the data repository via the NFS network.

23. The method of claim 3, wherein the data repository resides in multiple computers.

24. The method of claim 3, wherein the found file segment is stored in a segment library having a two-level directory structure.

25. The method of claim 24, wherein a file name of the found file segment includes a hexadecimal digit sequence representative of a portion of a memory address of the found file segment.

26. The method of claim 24, wherein a directory name of a directory containing the found file segment includes a hexadecimal digit sequence representative of a portion of a memory address of the found file segment.

27. The method of claim 3, wherein checkpoints lock at least a portion of the data repository during a file system copy.

28. The method of claim 27, wherein at least one species of objects of the file segments is locked independently of another species of objects of the file segments.

29. The method of claim 3, wherein the application program and database are associated with a trade order management system (TOMS).

30. A system for an application program to manage memory associated with an object-oriented database (OODB) accessed by the application program, the application program associated with an operating system, the method comprising:

(a) means for connecting to a data repository of the database, the data repository including a plurality of memory-mapped file segments stored on at least one nonvolatile memory medium;

(b) means for registering a fault handler with the operating system, the fault handler associated with the data repository;

(c) means for catching, by the fault handler, a segmentation fault issued for an object referenced by the application program and resident in the data repository, the segmentation fault issued at an interrupt location in the application program;

(d) means for finding a file segment of the data repository that corresponds to the referenced object;

(e) means for mapping the found file segment into a main memory; and

(f) means for restarting the application program at the interrupt location.

31. The system of claim 30, wherein the application program is a portfolio management application, and the database is a portfolio management database.

32. The system of claim 30, wherein the data in the data repository is not copied into the main memory from the data repository when needed by the application program and is instead directly accessed by the application program.

33. An object-oriented database (OODB), comprising:

(a) a data repository including a plurality of memory-mapped file segments stored on at least one nonvolatile memory medium;

(b) the file segments including objects directly interconnected by memory pointers;

(c) each object having an associated stored time stamp, the time stamp indicating at least one of a time when the object first appeared in the data repository and a time when the object became invalid; and

(d) the data in the data repository not being copied into a main memory of a computer accessing the data,

wherein the data repository has an associated fault handler, the fault handler registered by an application program with an operating system associated with the application program,

wherein the fault handler is configured to catch a segmentation fault issued for an object referenced by the application program and resident in the data repository, the segmentation fault issued at an interrupt location in the application program,

wherein a file segment of the data repository that corresponds to the referenced object is mapped into the main memory, and

wherein the application program is restarted at the interrupt location.

34. The database of claim 33, wherein the application program is a portfolio management application, and the data repository is a portfolio management data repository.

35. A machine-readable medium encoded with a plurality of processor-executable instructions for:

(a) connecting to a data repository of an object-oriented database (OODB) accessed by an application program, the application program associated with an operating system, the data repository including a plurality of memory-mapped file segments stored on at least one nonvolatile memory medium;

(b) registering a fault handler with the operating system, the fault handler associated with the data repository;

(c) catching, by the fault handler, a segmentation fault issued for an object referenced by the application program and resident in the data repository, the segmentation fault issued at an interrupt location in the application program;

(d) finding a file segment of the data repository that corresponds to the referenced object;

(e) mapping the found file segment into a main memory; and

(f) restarting the application program at the interrupt location.

36. The computer-readable medium of claim 35, wherein each object in the data repository has an associated stored time stamp, the time stamp indicating at least one of a time when the object first appeared in the data repository and a time when the object became invalid.